Profit Maximization of Big Data Jobs in Cloud Using Stochastic Optimization

Seyed Morteza Nabavinejad, and Maziar Goudarzi, *Senior Member, IEEE*

***Abstract*—Reserved instances offered by cloud providers make it possible to reserve resources and computing capacity for a specific period of time. One should pay for all the hours of that time interval; in exchange, the hourly rate is significantly lower than on-demand instances. Reserved Instances can significantly reduce the monetary cost of resources needed to process big data applications in cloud. However, purchases of these instances are non-refundable, and hence, one should be able to estimate the required resources prior to purchase to avoid over-payment. It becomes important especially when the results obtained by big data job has monetary value, such as business intelligence applications. But, estimating the resource demand of big data processing jobs is hard because of numerous factors that affect them such as data locality, data skew, stragglers, internal settings of big data processing framework, interference among instances, instances availability, etc. To maximize the profit of processing such big data jobs in cloud considering fluctuating nature of their resource demand, as well as reserved instances limitations, we propose Reserved Instances Stochastic Allocation (RISA) approach. Using historical traces of resource demand of big data jobs submitted by user, RISA leverages stochastic optimization to determine the amount of resources needed to be reserved for that user to maximize the profit. Our evaluation using real-world traces shows that RISA can increase the net profit by up to 10x, compared to previous approaches. RISA can also find solutions as close as 2% to the best possible solution.**

***Index Terms*—Cloud Computing, Big Data Processing, Reserved Instances, Stochastic Optimization, Profit Maximization**

I. Introduction

NOWADAYS, executing big data applications in cloud is easier than ever. With the dedicated services offered by cloud providers for big data processing such as Amazon EMR [1], customers can deploy their applications on pre-configured clusters as fast as possible. Moreover, the Reserved Instances (RIs) offered by cloud provide discounted prices compared to on-demand instances [2], and hence, can potentially reduce the monetary cost of big data applications, specifically the periodic ones [3], [4].

A Reserved Instance is a reservation of computing resources of cloud for 1-year term or 3-year term. While a user should pay for all of the hours of term, the hourly rate is less than conventional on-demand instances and offers up to 75% discount. The RIs can make the execution of big data jobs cheaper, provided that the challenges that arise from nature of RIs and big data are addressed in advance. One of the challenges of using RIs is that they are non-refundable, and hence, the customer cannot return them after purchase. While for some types of RIs such as standard reserved instances the customers can resell them in RIs Market [5], most of RIs including Scheduled Reserved Instances have not this feature. Considering this fact, along with the long term contracts of purchasing RIs (minimum one year), it is crucial to choose the number of instances carefully to avoid extra costs due to inaccurate estimation of resource demand.

In addition to aforementioned challenge of RIs, several inherent features of big data processing applications such as data locality [6], [7], data skew [8]–[10], stragglers [11], [12], and internal settings of big data processing frameworks [13], [14] cause significant fluctuation in the performance, and consequently, resource demand of jobs. The previous works that study the real world traces of big data jobs [15], [16], also report the fluctuation in the resource demand. Hence, it becomes even harder to determine the exact number of VM instances needed to finish the application before a specific deadline or obtain a desired throughput.

Previous approaches that allocate the resources to a job (and not a user) either consider the average resource demand [17]–[19] of the job or conservatively the upper bound [20]–[22]. Since none of these approaches take into account the resource demand variation of jobs of a user, they are unable to fully leverage the discounted prices of RIs to maximize the profit. Several approaches [23]–[26] that consider the RIs, focus on outdated light, medium, and heavy RIs that are not offered by cloud providers anymore. Hence, these approaches are not applicable to current clouds. Another disadvantage of some of the previous works such as [25], [27] is that they need resource demand of jobs in an hourly granularity, which is very hard to obtain accurately, if not possible.

To improve the profit of processing big data applications on RIs, it is essential to tackle the aforementioned challenges, i.e., non-refundable long term contracts and fluctuating resource demand of big data jobs. In this paper we propose RISA (Reserved Instances Stochastic Allocation) approach which uses stochastic optimization to find the best number of RIs to be rented to maximize the profit of processing big data jobs over the course of time. RISA derives the distribution of resource demand of jobs submitted by user through resource demand histories. Then, it employs the News Vendor Problem, a well-known stochastic optimization problem, to allocate RIs to users. Comparing RISA against other methods indicate that it can improve the net profit by up to 10x compared to a conservative approach, and achieve results as close as 2% to the optimal result.

Seyed Morteza Nabavinejad is with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. Part of this research is done while he was a Ph.D. student at Sharif University of Technology.
E-mail: nabavinejad@ipm.ir

Maziar Goudarzi is with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
E-mail: goudarzi@sharif.edu

The key contributions we make in this paper is as follows:

- To the best of our knowledge, RISA is the first work that exploits stochastic optimization in the form of News Vendor Problem for allocating RIs to maximize the profit.
- RISA exploits historical traces of resource usage of jobs to extract the probability distribution of their resource demand, instead of employing point estimates (e.g., mean of historical traces) when allocating resources.
- RISA considers non-refundable long term contracts of RIs and fluctuating resource demand of big data jobs when allocating the resources and tries to maximize the net profit of users when employing RIs.
- We exploit real-world Hadoop traces to motivate RISA. Moreover, we employ the same traces, as well as synthetic ones, to evaluate the effectiveness of RISA under different situations such as various probability distribution for resource demand.

The rest of the paper is organized as follows. In section II we demonstrate the resource demand variation of big data jobs and introduce the reserved instances and news vendor problem, which are necessary background information to follow the paper. In section III we formulate the problem and present RISA to tackle it using stochastic optimization. In section IV we evaluate the performance of RISA. We discuss the related work in sectoin V and in section VI we conclude this paper and provide directions for future work.

II. Motivation and Background

A. Resource Demand Variation

To show the significant fluctuation of resource demand of big data jobs submitted by the same user, we employ the OpenCloud Hadoop cluster trace [28]. This trace consists of logs of MapReduce jobs submitted by individuals at CMU. While the username of jobs are anonymized, it is still possible to identify the jobs submitted by same user. There is no information in the trace regarding the number of slots (resource allocation unit in Hadoop v1) or nodes allocated to each job. Hence, we estimate the resource demand of each job using number of its tasks. Each task needs one slot to process, and so we can consider the number of slots of each job equivalent to the number of its tasks.

Depicting the resource demand of jobs of 40 users in Fig. 1 reveals a significant variation. For each user, We see a wide gap between minimum and maximum resource demand of its jobs; note that the horizontal lines of each box in Fig. 1 show the 25^{th}, 50^{th}, and 75^{th} percentile of dots (i.e., resource demand of jobs). The average resource demand variation of jobs for the users depicted in Fig. 1 is around 156% and the maximum is 1292%. We use (1) for calculating the variation. When deciding to employ the RIs, a cloud user should consider this significant variation to maximize the profit.

$$variation = \frac{\sigma}{\mu} * 100 \tag{1}$$

B. Reserved Instances

In response to diversity of resource demand of cloud users, cloud providers offer various types of instances such as on-demand [29], spot [30], and reserved instances [2]. Reserved Instances (RI) can help save money while maintaining flexibility. They also provide a capacity reservation under specific conditions. Amazon EC2 [31] offers RIs in three categories: 1) Standard RIs: offer up to 75% discount compared to on-demand instances and are suitable for steady-state usage. 2) Convertible RIs: similar to standard RIs, but with less discount (up to 54%) and more flexibility. 3) Scheduled RIs: suitable for periodic jobs that are launched at a specific time such as calculating value at risk of a bank every weekday afternoon. They provide up to 10% discount.

The discounted price of RIs can incentivize their employment by cloud customers. However, the customers should be aware of long term contracts of purchasing RIs (minimum one year). Moreover, the RIs are non-refundable and the customers should pay for the whole duration of contract. The fluctuating resource demand of jobs (as in Fig. 1) along with aforementioned limitations of RIs, make it challenging to choose the best number of RIs for a specific cloud user.

C. News Vendor Problem

The News Vendor Problem (NVP) is a well-known problem in stochastic optimization. The problem describes a news

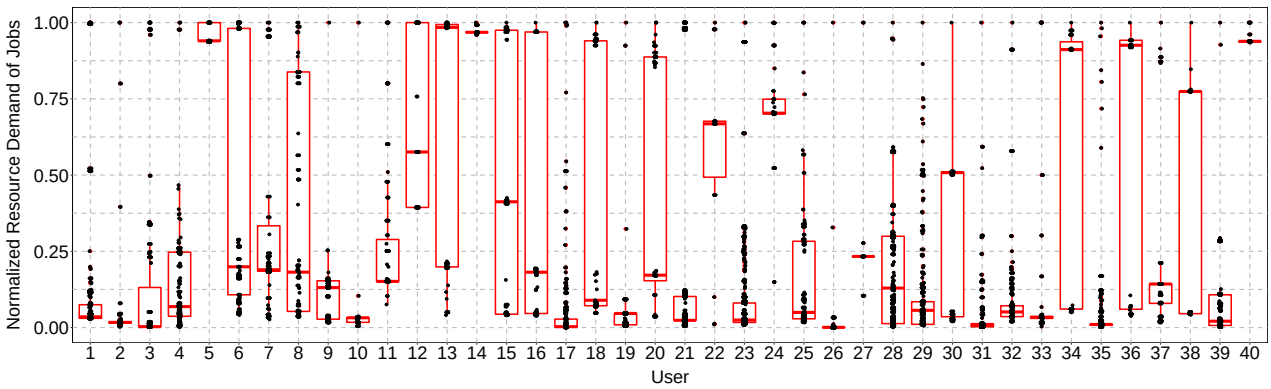


Fig. 1. Resource demand variation of MapReduce jobs submitted to OpenCloud Hadoop cluster by different users

vendor that every morning buys x newspapers, at a price of c per paper, from publisher. The maximum number of papers that vendor can buy is limited to u that shows either the purchase power of vendor or a limit set by publisher. The vendor tries to sell as many newspaper as possible at the price q. At the end of the day, the vendor can return the unsold papers to publisher at the return price r, where $r < c$. It is assumed that vendor cannot return to publisher during the day and buy more newspaper because the other vendors would have taken them.

In this problem, the goal is to determine the number of newspapers that the vendor should buy every day to maximize the profit over the course of time. Obviously, the newspaper demand varies from day to day and it is described by random variable ξ. Defining the y as the number of newspaper sold, w as the number of newspapers returned, and E_ξ as the expected value of ξ, the vendor's profit can be formulated as follow:

$$\begin{aligned} max \quad & \mathscr{Q}(x) - cx, \\ & 0 \leq x \leq u \end{aligned} \tag{2}$$

where

$$\mathscr{Q}(x) = E_\xi Q(x, \xi) \tag{3}$$

and

$$\begin{aligned} Q(x,\xi) = max \quad & qy(\xi) + rw(\xi) \\ s.t. \quad & y(\xi) \leq \xi, \\ & y(\xi) + w(\xi) \leq x, \\ & y(\xi), w(\xi) \geq 0 \end{aligned} \tag{4}$$

in the above equations, $\mathscr{Q}(x)$ represents the expected profit on return and sale, while $Q(x, \xi)$ indicates the same profit but when the demand for newspaper is at level ξ.

III. Proposed Approach

A. Problem Statement and Formulation

For a given user, we have a probability distribution for resource demand of jobs submitted by that user (via the job history of each user) in the form of number of instances. We also know the cost of each instance per year (considering 1-year term contracts) and the gross profit can be obtained from each instance. Now, we want to determine the number of RIs that we should employ to maximize the net profit of user over the course of time. Note that we cannot change the number of RIs after we made the decision, because of the non-refundable nature of RIs, and we should pay for all of them. If the resource demand of a job is more than the number of determined RIs, then the extra demand should be covered by on-demand instances to make sure that every job is received its required resources. Hence, considering the higher price of on-demand instances compared to RIs, when determining the number of RIs, is necessary.

In other words, the reserved instances should be employed for coarse-grain granularity, e.g., yearly basis, to help covering a large portion of computation with lower cost. In this case, considering the variation of job demand within a year can help to choose the number of RIs more wisely. On the other hand, The on-demand instances can address the fluctuation of the resource demand of jobs in finer granularity, e.g., on a hourly or daily basis.

TABLE I
Notation Used in the Paper

Parameter	Definition
NJ	Number of Jobs of User
IP	Instance Profit (per year)
ICR	Monetary Cost of One Reserved Instance (per year)
ICO	Monetary Cost of One On-Demand Instance (per year)
AIC	Average Cost of Instance (On-Demand and Reserved)
TRI	Total Number of Instances to be Reserved
TOI	Total Number of On-Demand Instances
IP/AIC	Ratio of Instance Profit to Average Cost of Instance
IDJ_i	Number of Instances Demanded by Job i
RDPD	Probability Distribution of Resource Demand of Jobs
RVD	A Random Variable that Follows RDPD
TGP	Total Gross Profit
TIC	Total Monetary Cost of Instance
TNP	Total Net Profit

In the NVP, the goal is to maximize the profit over the course of time. Hence, when the number of newspapers that is needed to be purchased is determined, it is supposed that the news vendor does not change the number of newspapers from day to day, but sticks to the predetermined one to maximize the profit in long term. It is similar to reserving the RIs for a long period of time (e.g., one year). However, there is a difference between NVP and our problem in this paper. In NVP once the vendor purchased the newspapers, he/she cannot return to whole seller to buy more if needed. But in our problem it is possible to cover the extra demand of job, which is beyond the number of RIs, by on-demand instances.

Before presenting the problem formulation, we describe all the parameters used in the rest of the paper in Table I.

The objective function of our problem is to maximize the total net profit for the user by deciding how many RIs to buy for the year ahead:

$$Maximize \quad TNP \tag{5}$$

For obtaining the net profit, we need to have the gross profit and cost of VMs. Hence, we have:

$$TNP = TGP - TIC \tag{6}$$

We can calculate the total cost of VMs by multiplying the cost of each instance per year and the number of RIs we decide to employ plus the price of on-demand instances required to cover the requests with resource demand beyond the available RIs .

$$TIC = TRI \times ICR + \sum_{i=1}^{NJ} max(IDJ_i - TRI, 0) \times (\frac{ICO}{NJ}) \quad (7)$$

In our work, we assume that user does not execute several jobs at the same time, but executes all of them sequentially. In other words, we consider non-overlapping jobs. Hence, we have considered the resource demand of one job (IDJ_i) in (7). However, for the cases where the user intends to execute several jobs at the same time, for example M jobs, the sum of the resource demand of those M jobs ($\sum_{i=1}^{M} IDJ_i$) should be considered in (7) instead of a single job. Otherwise, too many jobs would share one instance.

When we employ a reserved instance, we should pay the total cost for one year (ICR). However, for on-demand instances, we should pay only for the fraction of time that we employ the instance, i.e., the runtime of the job that is using the instance. Since we do not have the runtime of the jobs in advance, we divide the ICO by NJ (ICO/NJ) to find the average cost of on-demand instance per job, instead of considering the cost per year (ICO).

Since all the required instances of a job are either covered solely by RIs or a combination of RIs and on-demand instances, We can obtain the total gross profit by simply multiplying the demand of each job and the expected profit of each demand (IP).

$$TGP = \sum_{i=1}^{NJ} IDJ_i \times IP \quad (8)$$

B. Reserved Instances Stochastic Optimization

In this section, we discuss our RIs allocation approach to maximize the profit of big data jobs. Fig. 2 presents the overall flow of the *RISA*. The problem that *RISA* wants to address is choosing the number of rented RIs to maximize the profit over the course of time. Parts with colored backgrounds are the contributions of *RISA*.

Previous works that have studied the reserved instances, can be categorized into two groups: 1) The ones that focus on finding the proper number of reserved instances [23], [32] and 2) The ones that try to procure the right instance type, in addition to the number of instances [25], [27]. RISA, similar to the first category, focuses on finding the right number of instances to maximize the profit. So, it is more concerned with cost and profit of instances, rather than their configuration. We suppose that the cloud user already knows the best configuration for its jobs, based on historical information, and only needs to reserve the right number of instances. The user can choose from different configuration offered by cloud provider, similar to the ones listed in Table II, that fits its jobs resource demand better than others. It is similar to the "Immidiate Execution" model introduced by Cura [20].

According to Cura [20], there are three operational models for cloud: *immediate execution*, *delayed start*, and *cloud managed* (please see Related Work section for more details). In this paper, similar to [23], we consider the first model, i.e., immediate execution. Our goal is to satisfy all the required number of instances of a job either by RIs or on-demand instances, while maximizing the profit. In this model, immediate execution, user specifies the number of required instances of a job, type of instances, paying the monetary cost, configuration of jobs, SLO, etc.

In our approach, we assume that the expected revenue or profit per RI (*IP*) is known in advance via analyzing the history of user. The cost of RI and its corresponding on-demand instance (instance with the same configuration as RI but with on-demand price) is also known through cloud provider. Using the history of jobs submitted by the user, *RISA* fits a probability distribution (e.g., normal, uniform, lognormal) to resource demand (IDJ_i) of jobs to find their demand distribution (Jobs Demand Distribution in Fig. 2). It is one of the advantages of *RISA* over previous approaches, which use simple point estimates when allocating the resources such as [19], [20], [33] or the ones that only consider the recent history of jobs demand in their model such as RIPAM [23]. Having the probability distribution of resource demand, the cost of instances and expected revenue, *RISA* employs the NVP to solve the problem of choosing the number of RIs to be rented. By employing the NVP, *RISA* is able to find a balance between the gross profit that user can obtain and the monetary cost of RIs that should be paid. Since returning the idle instances is impossible due to non-refundable nature of RIs, the value of r (return price in NVP) is zero in our problem.

Having $r = 0$, we have the expected profit of reserved RIs as follow:

$$E[TNP] = IDJ_i \times IP - (TRI \times ICR + E[max(RVD - TRI, 0)] \times (\frac{ICO}{NJ})) \quad (9)$$

Eq. (9) is an extension of the newsvendor problem, known as inventory optimization. The objective is to minimize the cost of expected shortage quantity (term E[max(RVD-TRI,0)] in Eq. (9)) that leads to revenue lost, and consequently, maximizing the profit. In our problem, the expected shortage quantity shows the cost of on-demand instances that should be employed due to lack of RIs, which decreases the net profit. This problem, can be solved using the critical fractile formula

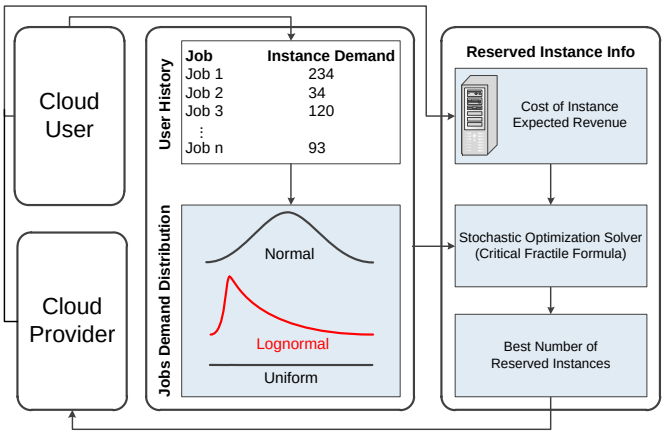


Fig. 2. Overall flow of RISA

TABLE II
INSTANCE CONFIGURATIONS USED IN EXPERIMENTS

	Instance Type	vCPU	Mem (GB)	All Upfront Price - ICR ($/Year)	On-Demand Price - ICO ($/Year)	Expected Revenue - IP ($)
Config 1	r5.2xlarge	8	32	2596	4415	5192
Config 2	m5.2xlarge	8	32	2006	3363	5015
Config 3	m4.2xlarge	8	32	2026	3504	6078
Config 4	c5.2xlarge	8	16	1764	2978	6174
Config 5	c4.2xlarge	8	15	2078	3486	8312
Config 6	h1.2xlarge	8	32	2601	4099	11704
Config 7	i3.2xlarge	8	61	3482	5466	17410
Config 8	d2.2xlarge	8	61	5904	12088	32472

the same as standard newsvendor problem. The only difference is that in newsvendor problem we have a maximization problem (maximizing net profit), but here we have a minimization problem (minimizing the cost of shortage quantity). However, the optimal q (number of newspapers or number of RIs in our work) would be the same for both problems [34], [35]. To solve (9) and find the optimal value of *TRI* to maximize the profit, *RISA* employs the critical fractile formula [36]:

$$TRI = F^{-1}(\frac{IP - AIC}{IP}) \tag{10}$$

F^{-1} stands for inverse cumulative distribution function of *RDPD*. The critical fractile (ratio in (10)) tries to balance the cost of over-provisioning (cost of idle instances) and under-provisioning (losing the profit of satisfying all demands). Note that the original form of ratio in (10) is $\frac{IP-AIC}{IP-r}$, but as mentioned earlier we have $r = 0$. In (10), AIC is the average cost of reserved and on-demand instances of a specific configuration. As mentioned before, since the *RISA* needs to cover the demand of job that exceeds the RIs by on-demand instances, we use AIC to inform *RISA* about the increased cost of on-demand instances compared to RIs. AIC is calculated as (11) where ORR is the ratio of on-demand price to reserved price of a specific instance configuration ($ORR = \frac{ICO}{ICR}$).

$$AIC = \frac{ICR + (ICO \times ORR)}{1 + ORR} \tag{11}$$

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

Instance configurations. The specification of RIs that we have used in the experiments are presented in Table II. The specifications are obtained from Amazon EC2 RIs [37]. The price are for standard 1-year term Linux instances with All Upfront payment option and corresponding on-demand instance. Since there is no information for expected revenue in the Hadoop cluster trace, we consider it as a multiple of instance price (from 2 for Config 1 to 5.5 for Config 8 with 0.5 steps). We consider different expected revenue for configurations to study its impact on performance of *RISA* in experimental results.

Systems compared. We compare RISA with the following approaches:

- *On-Demand*: As it names state, On-Demand does not reserve any instances, and only uses on-demand ones. We use it as a baseline approach in our experiments to show the value of RIs.
- *RIPAM*: Reserved Instance Provisioning strategy based on Autoregressive Model (RIPAM) [23] employs an autoregressive model to obtain the required number of RIs for average computation.
- *Conservative*: Some approaches [20]–[22] use conservative over-provisioning to satisfy all the resource demands. Conservative represents these approaches by renting RIs equal to maximum resource demand of jobs($TRI = max(IDJ_i)$). Conservative hopes to maximize the *TNP* by satisfying all the resource demands with RIs. However, when the resource demand is less than the maximum, Conservative renders the *TNP* low.
- *Optimal*: It is another baseline approach to represent the optimal profit obtained if the cloud tenant has perfect information of resource demands of future jobs that are going to be submitted by the user. Optimal implements a brute-force algorithm that examines all the possible combinations of RIs and on-demand instances to find maximum possible profit. We use it to show how much RISA solutions are close to best possible ones.

Workloads. We use workloads derived from OpenCloud Hadoop cluster trace [28] in our experiments. Each workload is for one user with different number of submitted jobs (from 86 to 159). To calculate the resource demand of each job (in the form of number of instances), we first obtained the number of slots required by that job similar to section II-A. The jobs with

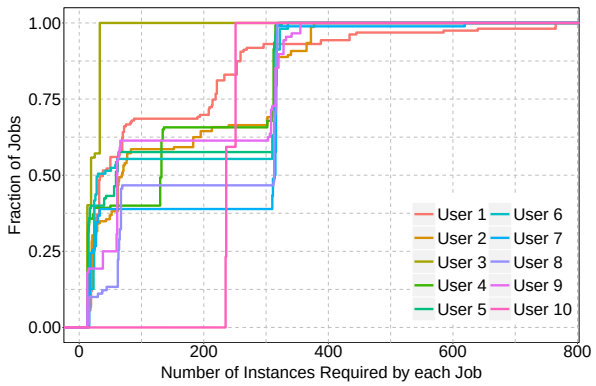


Fig. 3. Distribution of instance demand of jobs of each user.

TABLE III
WORKLOADS FROM OPENCLOUD HADOOP CLUSTER TRACE [28]

	Number of Jobs	Lognormal Distribution of Resource Demand (μ, σ)	Resource Demand Variation
User 1	159	(4.104 , 1.14166)	127%
User 2	152	(4.36457 , 1.23772)	93%
User 3	147	(3.03246 , 0.427717)	40%
User 4	140	(4.30553 , 1.36069)	86%
User 5	125	(4.21809 , 1.4066)	95%
User 6	103	(4.32779 , 1.31216)	93%
User 7	90	(4.67843 , 1.38348)	73%
User 8	90	(4.85342 , 1.0466)	68%
User 9	88	(4.42949 , 1.20188)	90%
User 10	86	(5.48792 , 0.031314)	3%

resource demand less than 100 slots are filtered out. Then, we divided the number of slots by the number of vCPU of instance configurations to have the number of required instances per job. Similar to previous works [38], [39], we considered one Map/Reduce slot per vCPU for each instance.

Number of jobs of each user and their resource demand distribution are presented in Table III. Distribution of instance demand of jobs of each user is shown in Fig. 3. We used *Distribution Fitting App of MATLAB* [40] and tried various probability distributions to find the most suitable one for resource demand of jobs with least amount of error, and lognormal was a better fit although none of the distributions we tried were perfect. If the resource demand of jobs of a user are that random that no suitable distribution can be found for it, then *RISA* is not applicable for that user.

B. Profitability and Cost Analysis

In this section, we discuss the main results obtained from the experiments. First, we report the number of RIs selected to be reserved by approaches under different instance configurations. As presented in Table II, while all the configurations have the same number of vCPUs, and consequently, the same number of slots, their *ICR*, *ICO*, and *IP* are different. Since *RIPAM* and *Conservative* approaches are unaware of impact of these two factors on the final profit of resource allocation, their number of reserved RIs is constant, regardless of instance configurations. However, *RISA* selects the number of reserved instances based on critical fractile introduced in (10) which is a function of *AIC* and *IP*, and hence, its number of RIs varies for different instance configurations. Optimal also considers the aforementioned factors and has different number of RIs for different configurations.

Fig. 4 depicts the number of RIs reserved (*TRI*) by each approach for different users under various instance configurations and Fig. 5 shows the number of on-demand instances (*TOI*) employed by each approach. Since the number of RIs for *Conservative* and *RIPAM* is constant for all the configurations, only one curve is plotted for each of them in Fig. 4. The same is true for *On-Demand* and *RIPAM* in Fig. 5. *On-Demand* does not use RIs at all, so there is no curve for it in Fig. 4. Also in Fig. 5, there is no curve for *Conservative* since it does not use on-demand instances. As can bee seen, *Conservative* has

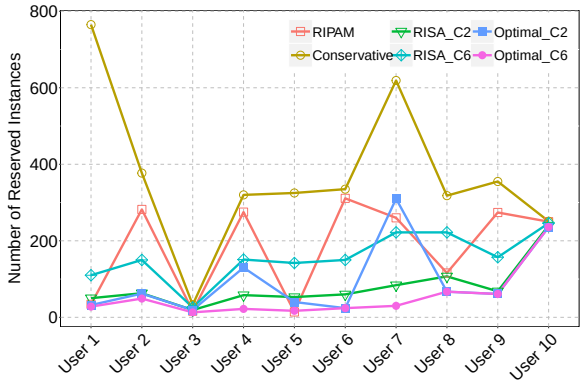


Fig. 4. Number of instances reserved by each approach for different users under different configurations. *RIPAM* and *Conservative* approaches have the same number of instances for different configurations, so there is only one curve for each of them.

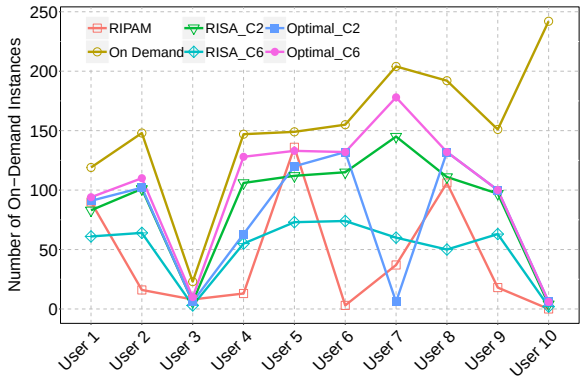


Fig. 5. Number of on-demand instances employed by each approach for different users under different configurations. *RIPAM* and *On-Demand* approaches have the same number of instances for different configurations, so there is only one curve for each of them.

the highest *TRI* for all the users, while *On-Demand* has the highest *TOI*, as expected from their definition. But, *RIPAM* has relatively lower number of RIs compared to *Conservative* and lower on-demand instances. Finally, we can see the sensitivity of *RISA* to *AIC* and IP of configurations. When the ratio of *IP* to *AIC* is low, the *TRI* of *RISA* is lower than that of

RIPAM, while its *TOI* is higher than it. However, when the IP/AIC ratio is on the increase, we see that *RISA* tends to reserve more RIs to gain more profit and approaches the *TRI* of *Conservative* and RIPAM, and uses less on-demand instances (lower *TOI*). Finally, considering both Fig. 4 and Fig. 5, we see that the number of RIs of *RISA* has the same trend as the Optimal approach.

After discussing the number of RIs and on-demand instances, we report the net profit obtained by each approach. The average net profit of all the users obtained by each approach under various configurations is shown in Fig. 6. As can be seen, the net profit of *RISA* always surpasses those of *RIPAM* and *Conservative*, and it is very close to optimal. The narrowest gap between *RISA* and *Optimal* is 1.5% in Config 1 and Config 2, and the widest gap is around 5% in Config 6 and Config 7. The *RISA* improves the net profit compared to *On-Demand*, *RIPAM*, and *Conservative* by up to 22%, 21%, and 10x, respectively.

According to (6), the net profit can be calculated by having gross profit and cost of instances. Since all the approaches fulfill the demand of a job either by RIs or on-demand instances, all of them obtain the same amount of gross profit. Hence, in the following we report the results of cost of instances to analyze the behavior of approaches regarding it. The average cost of instances for jobs of all users under different configurations is presented in Fig. 7. As expected, *Conservative* has the highest cost of instances due to its high number of RIs. *RISA's* cost of instances is very close to Optimal, where it increases the cost by less than 1% compared to Optimal in Config 1 (around 5% on average for all the configurations). Comparing to *On-Demand*, *RIPAM*, and *Conservative*, *RISA* decreases cost of instances by up to 16%, 10%, and 38% , respectively. .

Finally, for two configurations (Config 2 and Config 6) we explore the performance of approaches for each individual user (instead of average of users). The net profit and cost of instances for all the users under different approaches for two aforementioned configurations are depicted in Fig. 8. We can see that the performance of approaches heavily depends on the distribution of resource demand of jobs, as well as

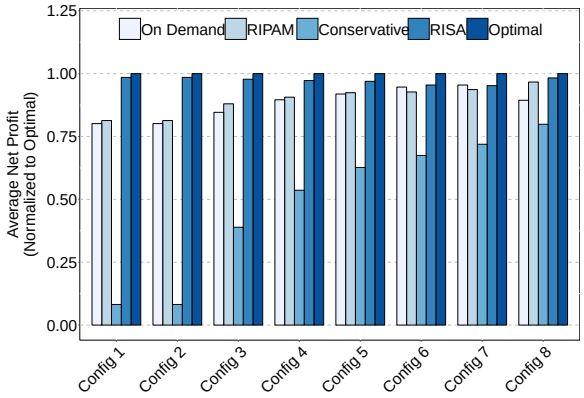


Fig. 6. Average net profit of all the users obtained by each approach under various configurations.

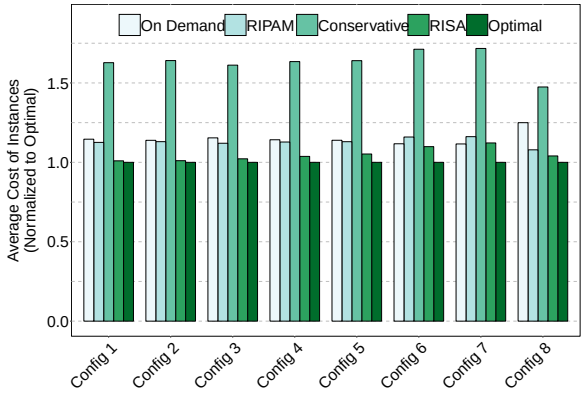


Fig. 7. Average cost of instances of all the users resulted by each approach under various configurations.

the instance configuration. For example, in Config 2 we see that *RISA* significantly outperforms *RIPAM* for User 2, User 4, User 6, and User 9 and yields more net profit. However, in results of Config 6, we see that the gap between *RISA* and *RIPAM* for the same users is not as wide as in Config 2. These results clearly show that how the IP/AIC ratio of different configurations affect the performance of approaches. Please note that for User 1 and User 2 in Config 2 and User 1 in Config 6, the net profit of *Conservative* is negative, and hence, we substituted them with zero to keep the figure consistent. The observations for cost of instances support the observations for net profit.

C. Resource Demand Coverage by RIs

The net profit has a direct relationship with the resource demand coverage by RIs. The more an approach covers the resource demand of jobs by RIs, the more net profit it can obtain because of difference between cost of RIs and cost of on-demand instances (Note that an approach must fulfill all the demand of job either fully by RIs or a combination of RIs and on-demands). Hence, in this section we report the resource demand coverage by RIs of different approaches for different configurations, i.e., what percentage of instances required by each job of a user are covered by RIs under various approaches. The results are shown in Fig. 9. As we depicted in Fig. 4 and mentioned in section IV-B, *RIPAM* and *Conservative* select the same number of RIs regardless of instance configurations. Hence, in Fig. 9, there is only one curve for each of them. However, for *RISA* and *Optimal* we have two curves, one for Config 2 and the other of Config 6. Please note that we have plotted the results of *RISA* and *Optimal* only for two configurations for the sake of readability of the figure.

As expected, *Conservative* satisfies all the demands completely because it considers the maximum demand when reserving RIs. The performance of *RIPAM* , *RISA*, and *Optimal*, however, depends on the distribution of instance demand of jobs of each user. Moreover, we see that the coverage percentage of *RISA* and *Optimal* changes with the instance configuration. For example in User 7, for Config 2 the average

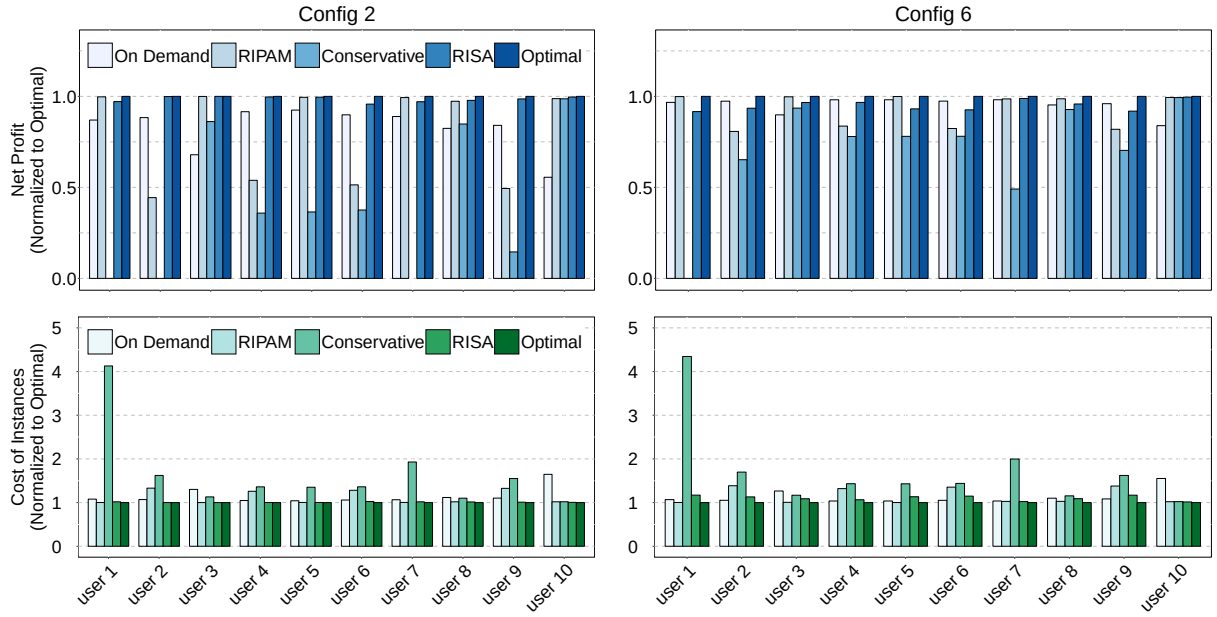


Fig. 8. Detailed results of all the users for two configurations.

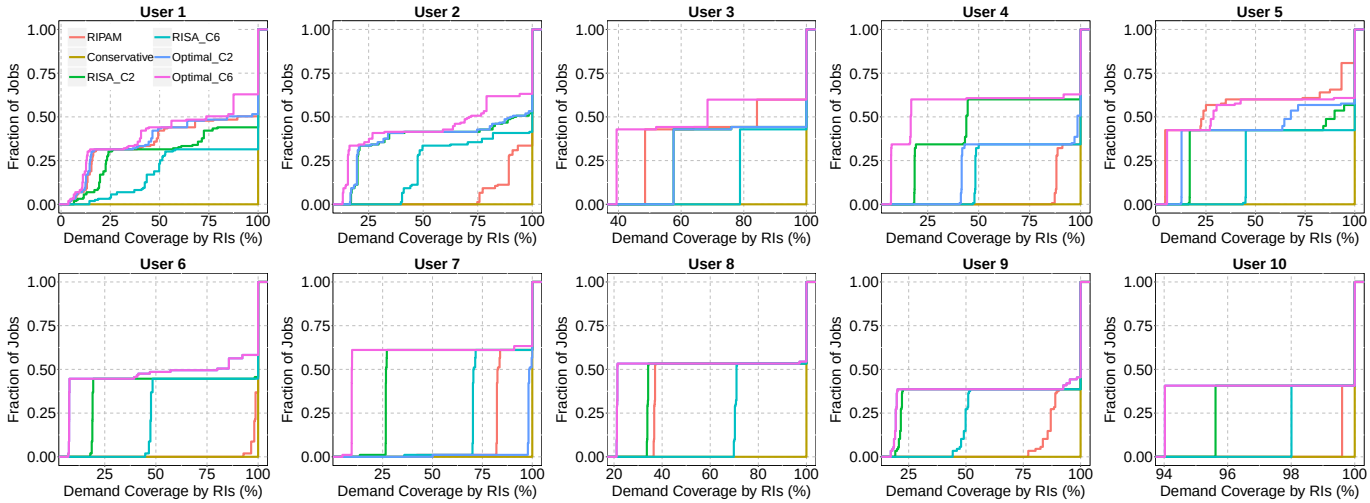


Fig. 9. Distribution of instance demand coverage of jobs by RIs for all users under different approaches.

coverage of *RISA* (55%) is significantly less than *RIPAM* (88%), but for Config 6 its coverage (81%) is approaching *RIPAM*. Please note that since on-demand does not employs RIs, there is no curve for it in Fig. 9.

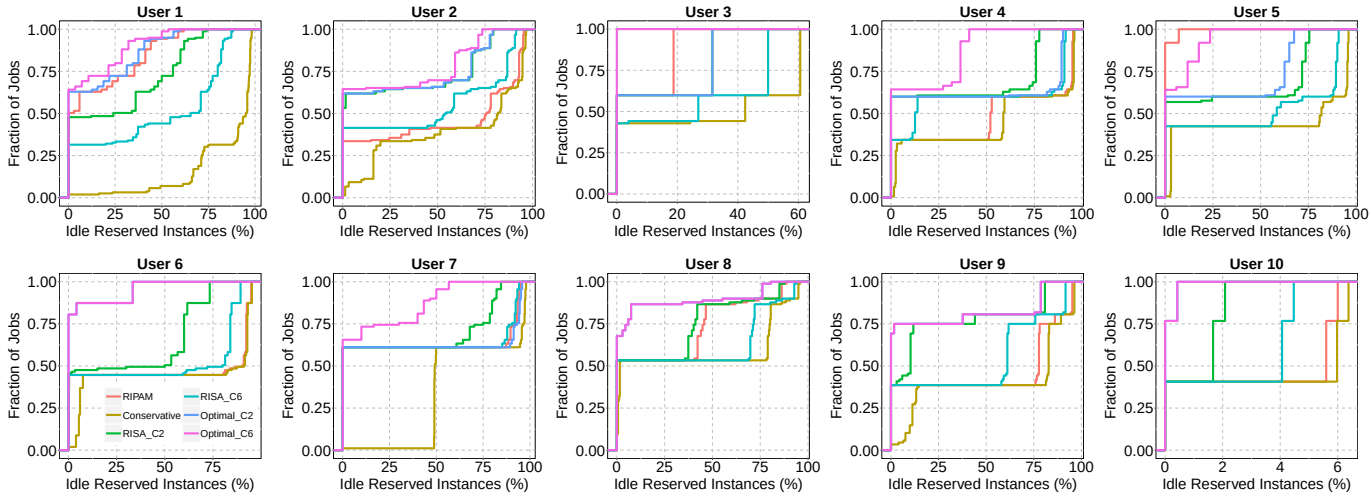
D. Resource Waste

In previous section we analyzed the instance demand coverage by RIs for different approaches. Now, we want to demonstrate why sometimes an approach e.g., *Conservative* has poor net profit despite its significant demand coverage. The key is idle RIs that waste the money without being used. Fig. 10 shows the distribution of idle RIs not being used in each job of users. As can be seen, *Conservative* suffers from significant amount of idle RIs. These idle RIs impose significant cost without being involved in gross profit, and hence, cause the net profit to decrease dramatically.

Considering Fig. 10 and Table III, we can see a clear relation between resource demand variation of users and resource waste. User 10 has the lowest resource demand variation (3%), and consequently, it has the lowest amount of idle RIs (less than 7% for all the jobs under different approaches). On the other hand, we see too much resource waste in User 1 that has 127% resource demand variation. The average amount of idle instances of *Conservative* for all the jobs in User 1 is around 84%. From the results, we can conclude that when the probability distribution of resource demand of jobs is narrow, we have a low amount of idle RIs. However, when the distribution is wide, we should expect high amount of idle RIs, regardless of resource allocation approach.

E. Sensitivity Analysis

To study the impact of demand distribution, we employed two synthetic workloads with uniform and normal instance



Fig

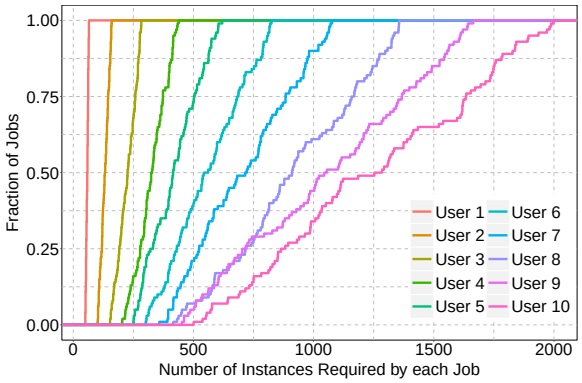


Fig. 11. Distribution of instance demand of jobs of each user in synthetic workload with uniform distribution.

demand distribution. Each workload consists of 10 users with 100 jobs per user. The probability distribution for instance demand of jobs of each user is presented in Table IV and distribution of instance demand of jobs of each user for uniform distribution and normal distribution workloads are presented in Fig. 11 and Fig. 12, respectively. Note that since the instance demand of a job should be positive, for normal distribution we have substituted the negative or zero numbers generated by random number generator function with positive ones.

In synthetic workloads, we consider different distribution compactness for each user, so they can represent users with various demand patterns in real world. For uniform distribution, the difference between two boundaries (i.e., a and b) becomes wider for each user. For User 1, we have the narrowest distribution with $b = a \times 1.3$ and for User 10 we have the widest one with $b = a \times 4$. The ratio between a and b in other users between User 1 and User 10 increases by 0.3 steps. For normal distribution, again the User 1 has the narrowest distribution with $\sigma = \mu \times 0.1$ and User 10 has the widest one with $\sigma = \mu \times 1$.

The net profit yielded by each approach for different users under different configurations is presented in Fig. 13 and Fig. 14 for uniform and normal distributions, respectively. We can see the superiority of *RISA* over *RIPAM* and *Conservative*

TABLE IV
SPECIFICATION OF SYNTHETIC WORKLOADS WITH NORMAL AND UNIFORM DEMAND DISTRIBUTIONS

	Workload with Uniform Distribution (a , b)	Workload with Normal Distribution (μ, σ)
User 1	(50 , 65)	(50 , 5)
User 2	(100 , 160)	(100 , 20)
User 3	(150 , 285)	(150 , 45)
User 4	(200 , 440)	(200 , 80)
User 5	(250 , 625)	(250 , 125)
User 6	(300 , 840)	(300 , 180)
User 7	(350 , 1085)	(350 , 245)
User 8	(400 , 1360)	(400 , 320)
User 9	(450 , 1665)	(450 , 405)
User 10	(500 , 2000)	(500 , 500)

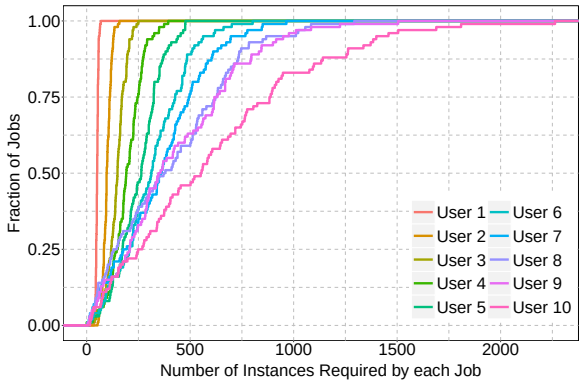


Fig. 12. Distribution of instance demand of jobs of each user in synthetic workload with normal distribution.

in both figures, which emphasizes the ability of *RISA* to maximize the net profit in different situations.

We can also see the impact of distribution compactness on the performance of approaches in Fig. 13 and Fig. 14. As expected, when the distribution is narrow (e.g., User 1), we see that the net profit of all the approaches is very close to each other. As distribution becomes wider, and consequently, the instance demand of jobs becomes more uncertain, we see

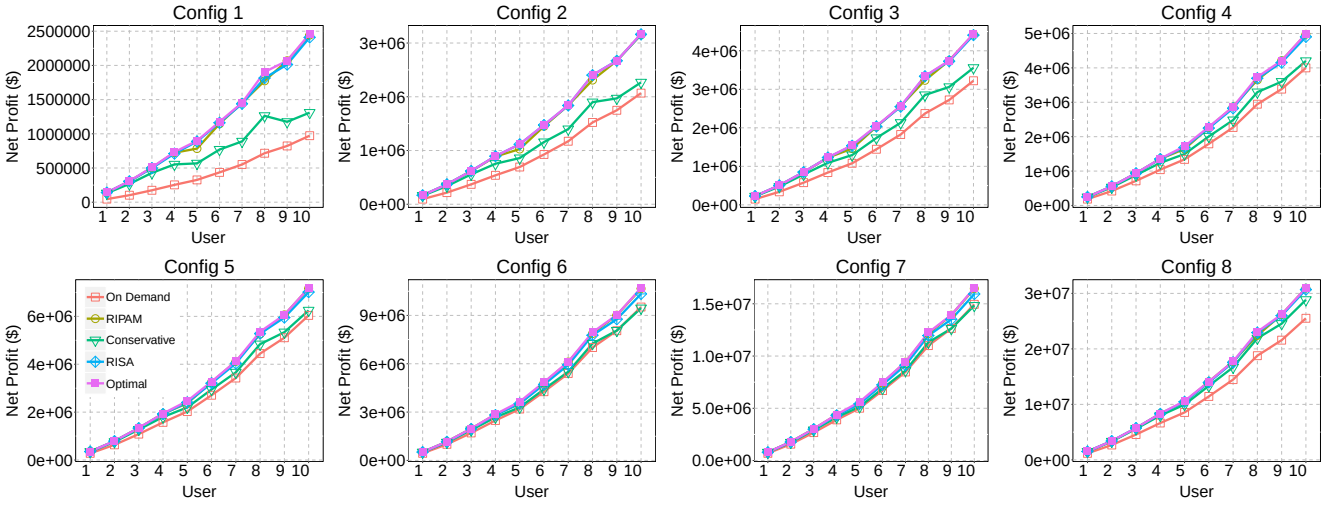


Fig. 13. Net profit obtained by each approach for all the users under various configurations for uniform demand distribution.

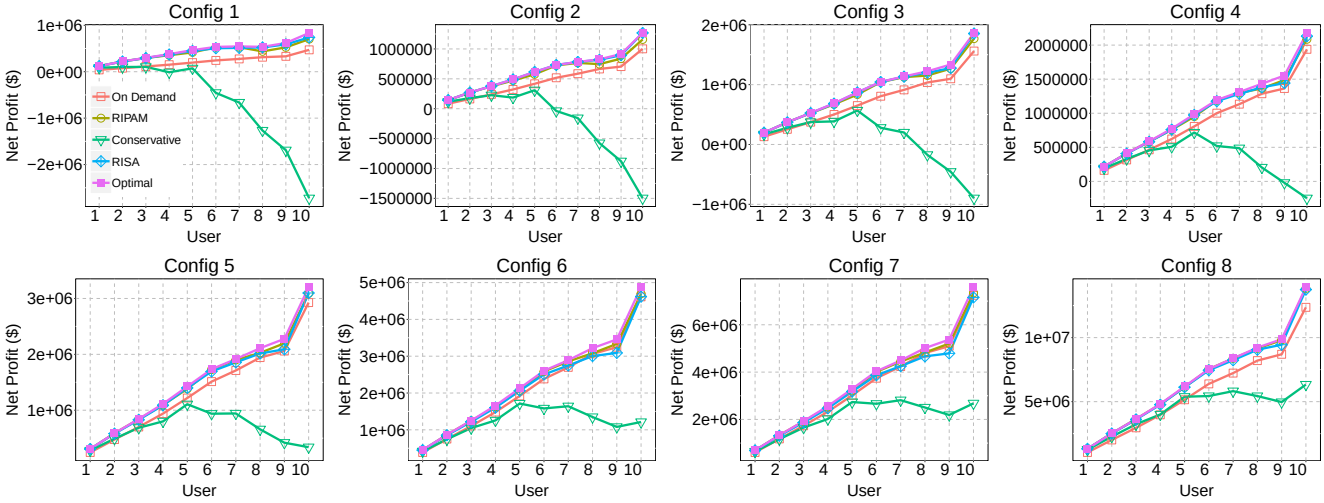


Fig. 14. Net profit obtained by each approach for all the users under various configurations for normal demand distribution.

that *RISA* gains more net profit than other approaches because it can successfully handle the uncertainty in resource demand of jobs through stochastic optimization. In both figures, the net profit of *RISA* is very close to *Optimal*, which shows the ability of *RISA* to find near optimal solutions. Finally, we see that the performance of *Conservative* improves as the instance configuration changes from Config 1 to Config 8 and the IP/AIC ratio of configurations increases.

V. Related Work

Proposed approaches for resource allocation and scheduling of big data processing applications in cloud and clusters have different targets such as deadline [41]–[44], cost [19], [20], [33], [45]–[47], and energy [38], [48]–[51].

Among the approaches focused on cost, Cura [20] introduces three operational models for cloud: *Immediate execution* where the user determines the VM instances needed for executing the job, and cloud provider only allocates them to user. In the second model, *delayed start*, the user still is responsible for determining the resources, but the cloud provider might allocate them to user with a delay to maximize its profit. However, cloud provider should consider the deadline of jobs and make sure they can meet them. Finally, in the third model which is called *Cloud managed*, cloud provider is responsible for resource provisioning and scheduling of jobs. The user only submits the job with its desired deadline. In this model, the profit of cloud provider can be maximized. Cura aims to increase the resource utilization of cloud by mixing a variety of MapReduce workloads, and hence, increase the profit of cloud provider based on the *Cloud managed* model.

Maximizing the profit of cloud providers when processing big data in federated clouds, is the goal of [33]. This work considers the various VM configurations with fluctuating costs, data transfer cost, and the deadline of arriving jobs. To reduce the VM and data transfer cost, they use FedSCD algorithm [52]. FedSCD leverages data locality and improves the utilization of resources by giving priority to the idle resources. They also reorder the arriving jobs to guarantee the completion of accepted jobs within their respective deadline to avoid penalty due to violating the deadlines.

All the aforementioned approaches consider an individual job, instead of all the jobs of a user, when allocating the resources. They also do not consider the RIs when allocating the resources, but the on-demand instances. Finally, they all rely on point estimates for resource demand of jobs, and ignore its stochastic nature.

Unlike the previous approaches, which use point estimates based on historical runtimes, 3Sigma [53] leverages distribution of runtime histories of relevant jobs to mitigate the runtime uncertainty when scheduling the jobs. 3Sigma considers runtime uncertainty, however it is proposed for physical clusters, and hence, does not deal with the challenges arises from nature of RIs.

Among the previous works that focus on reserved instances, Mazzucco et al. [54] aims to maximize the net revenue of cloud provider. It considers two types of customers where the premium ones will receive monetary compensation when there is not enough resources to execute their job, while the basic ones do not receive the monetary compensation and can only expect a best effort service. The net profit of cloud provider is calculated as the money it receives for server usage minus the energy cost and monetary compensation of premium customers. Unlike this approach, we aim to maximize the net profit of the cloud users.

Shen et al. [27] propose the CoH-R method to combine the on-demand and reserved instances. CoH-R needs the resource demand (or distribution of resource demand) of jobs in an hourly granularity to model the cost of instances. This approach, similar to several previous works such as [24], [25] considers the abolished light, medium, and heavy RIs model, which are not in use anymore. Hence, these approaches are not applicable to current RIs offered by cloud providers.

Ran et al. [23] propose the Reserved Instance Provisioning strategy based on Autoregressive Model (RIPAM) to determine the number of RIs. The proposed approach considers the difference between monetary cost of RIs and on-demand instances, but does not consider the profit can be obtained from each instance. Hence, unlike *RISA*, the RIPAM approach cannot distinguish between different instance configurations when reserving the RIs.

VI. Conclusion and Future Work

We presented RISA which uses stochastic optimization to maximize the profit of deploying big data jobs on RIs. RISA is aware of fluctuating nature of resource demand of big data jobs submitted by a user. It fits a probability distribution to resource demand of jobs and then uses robust analytical stochastic optimization method based on News Vendor Problem for resource allocation. Hence, it can obtain the highest profit over the course of time provided that the fitted probability distribution has little error. Using RISA on a wide collection of jobs from a real-world Hadoop cluster, as well as synthetic ones, indicated significant profit gain compared against other methods.

In this paper we focused on finding the proper number of RIs to be reserved, assuming that the instance type is predetermined by user. Considering various types of big data jobs such as CPU-intensive, Memory-intensive, and IO-intensive ones, it is an interesting direction for future research to determine the proper instance type for each user based on the resource demand of jobs, in addition to the number of instances. To achieve this goal, we need to incorporate multiple dimensions for distributions of different resources, such as memory and IO, instead of only considering the CPU consumption of the jobs. Furthermore, it is interesting to exploit the combination of several types of RIs for allocating to users, instead of one specific type.

In this work we consider the SLO as the number of instances that should be allocated to the job. We also assume that the job is executed in the form of a single wave of tasks. However, we can process big data jobs in multiple waves (e.g., multiple Map or Reduce waves) when the number of tasks is more than available slots, and hence, we cannot run all the tasks concurrently. If we consider the completion time of jobs as SLO, then finding the number of instances with least amount of cost to complete the job within its SLO, either with single-wave or multiple-wave execution, can be another direction for future work.

References

[1] "Amazon emr," https://aws.amazon.com/emr/, accessed Oct. 3th 2017.

[2] "Reserved instances," https://aws.amazon.com/ec2/pricing/reserved-instances/, accessed June. 22th 2018.

[3] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu, "Data warehousing and analytics infrastructure at facebook," in *Proceedings of the ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 1013–1020.

[4] Z. Ren, X. Xu, J. Wan, W. Shi, and M. Zhou, "Workload characterization on a production hadoop cluster: A case study on taobao," in *IEEE IISWC'12*. IEEE, 2012, pp. 3–13.

[5] "Ris market," http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ri-market-general.html, accessed Oct. 3th 2017.

[6] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality," *IEEE/ACM Trans. Networking*, vol. 24, no. 1, pp. 190–203, 2016.

[7] X. Ma, X. Fan, J. Liu, H. Jiang, and K. Peng, "vlocality: Revisiting data locality for mapreduce in virtualized clouds," *IEEE Network*, vol. 31, no. 1, pp. 28–35, 2017.

[8] Q. Chen, J. Yao, and Z. Xiao, "Libra: Lightweight data skew mitigation in mapreduce," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2520–2533, 2015.

[9] G. Liu, X. Zhu, J. Wang, D. Guo, W. Bao, and H. Guo, "Sp-partitioner: A novel partition method to handle intermediate data skew in spark streaming," *Future Generation Comp. Systems*, 2017.

[10] S. M. Nabavinejad and M. Goudarzi, "Data locality and vm interference aware mitigation of data skew in hadoop leveraging modern portfolio theory," in *SAC'18*. ACM, 2018, pp. 175–182.

[11] Q. Chen, C. Liu, and Z. Xiao, "Improving mapreduce performance using smart speculative execution strategy," *IEEE Trans. Computers*, vol. 63, no. 4, pp. 954–967, 2014.

[12] H. Xu and W. C. Lau, "Optimization for speculative execution in big data processing clusters," *IEEE Trans. Parallel and Distributed Systems*, vol. 28, no. 2, pp. 530–545, 2017.

[13] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of mapreduce programs," *VLDB Endowment*, vol. 4, no. 11, pp. 1111–1122, 2011.

[14] S. M. Nabavinejad, M. Goudarzi, and S. Mozaffari, "The memory challenge in reduce phase of mapreduce applications," *IEEE Trans. Big Data*, vol. 2, no. 4, pp. 380–386, 2016.

[15] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems a cross-industry study of mapreduce workloads," *VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.

[16] S. Kavulya, J. Tan, R. Gandhi, and P. Narasimhan, "An analysis of traces from a production mapreduce cluster," in *IEEE/ACM CCGrid'10*. IEEE Computer Society, 2010, pp. 94–103.

[17] A. Verma, L. Cherkasova, and R. H. Campbell, "Aria: automatic resource inference and allocation for mapreduce environments," in *Proceedings of the 8th ACM international conference on Autonomic computing*. ACM, 2011, pp. 235–244.

[18] S. M. Nabavinejad and M. Goudarzi, "Faster mapreduce computation on clouds through better performance estimation," *IEEE Trans. Cloud Computing*, 2017.

[19] X. Xu and M. Tang, "A new approach to the cloud-based heterogeneous mapreduce placement problem," *IEEE Trans. Services Computing*, vol. 9, no. 6, pp. 862–871, 2016.
[20] B. Palanisamy, A. Singh, and L. Liu, "Cost-effective resource provisioning for mapreduce in a cloud," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1265–1279, 2015.
[21] X. Xu and M. Tang, "A more efficient and effective heuristic algorithm for the mapreduce placement problem in cloud computing," in *IEEE CLOUD'14*. IEEE, 2014, pp. 264–271.
[22] L. Chen and H. Shen, "Consolidating complementary vms with spatial/temporal-awareness in cloud datacenters," in *Proceedings IEEE INFOCOM*. IEEE, 2014, pp. 1033–1041.
[23] Y. Ran, J. Yang, S. Zhang, and H. Xi, "Dynamic iaas computing resource provisioning strategy with qos constraint," *IEEE Trans. Services Computing*, vol. 10, no. 2, pp. 190–202, 2017.
[24] S. Niu, J. Zhai, X. Ma, X. Tang, W. Chen, and W. Zheng, "Building semi-elastic virtual clusters for cost-effective hpc cloud resource provisioning," *IEEE Trans. Parallel and Distributed Systems*, vol. 27, no. 7, pp. 1915–1928, 2016.
[25] D. Candeia, R. A. Santos, and R. Lopes, "Business-driven long-term capacity planning for saas applications," *IEEE Trans. Cloud Computing*, vol. 3, no. 3, pp. 290–303, 2015.
[26] W. Wang, B. Li, and B. Liang, "To reserve or not to reserve: Optimal online multi-instance acquisition in iaas clouds," in *Proceedings of the 10th International Conference on Autonomic Computing (ICAC'13)*, 2013, pp. 13–22.
[27] S. Shen, K. Deng, A. Iosup, and D. Epema, "Scheduling jobs in the cloud using on-demand and reserved instances," in *EuroPar'13*. Springer, 2013, pp. 242–254.
[28] K. Ren, Y. Kwon, M. Balazinska, and B. Howe, "Hadoop's adolescence: an analysis of hadoop usage in scientific workloads," *VLDB Endowment*, vol. 6, no. 10, pp. 853–864, 2013.
[29] "On-demand instances," https://aws.amazon.com/ec2/pricing/on-demand/, accessed June. 22th 2018.
[30] "Spot instances," https://aws.amazon.com/ec2/spot/, accessed June. 22th 2018.
[31] "Amazon ec2," https://aws.amazon.com/ec2/, accessed June. 22th 2018.
[32] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Iaas reserved contract procurement optimisation with load prediction," *Future Generation Computer Systems*, vol. 53, pp. 13–24, 2015.
[33] T. Gouasmi, W. Louati, and A. H. Kacem, "Geo-distributed bigdata processing for maximizing profit in federated clouds environment," in *26th Euromicro PDP*. IEEE, 2018, pp. 85–92.
[34] E. L. Porteus, "Stochastic inventory theory," *Handbooks in operations research and management science*, vol. 2, pp. 605–652, 1990.
[35] N. C. Petruzzi and M. Dada, "Pricing and the newsvendor problem: A review with extensions," *Operations research*, vol. 47, no. 2, pp. 183–194, 1999.
[36] T.-M. Choi, *Handbook of Newsvendor problems: Models, extensions and applications*. Springer, 2012, vol. 176.
[37] "Amazon ec2 reserved instances pricing," https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/, accessed Jun. 9th 2018.
[38] L. Mashayekhy, M. M. Nejad, D. Grosu, Q. Zhang, and W. Shi, "Energy-aware scheduling of mapreduce jobs for big data applications," *IEEE Trans. Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2720–2733, 2015.
[39] F. Yan, L. Cherkasova, Z. Zhang, and E. Smirni, "Optimizing power and performance trade-offs of mapreduce job processing with heterogeneous multi-core processors," in *IEEE CLOUD'04*. IEEE, 2014, pp. 240–247.
[40] "Distribution fitting app," MATLAB and Statistics Toolbox 8.2, The MathWorks, Inc., Natick, Massachusetts, United States.
[41] S. Dimopoulos, C. Krintz, and R. Wolski, "Pythia: Admission control for multi-framework, deadline-driven, big data workloads," in *IEEE CLOUD'17*. IEEE, 2017, pp. 488–495.
[42] C.-H. Chen, J.-W. Lin, and S.-Y. Kuo, "Mapreduce scheduling for deadline-constrained jobs in heterogeneous cloud computing systems," *IEEE Trans. Cloud Comp.*, vol. 6, no. 1, pp. 127–140, 2018.
[43] D. Cheng, J. Rao, C. Jiang, and X. Zhou, "Resource and deadline-aware job scheduling in dynamic hadoop clusters," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2015, pp. 956–965.
[44] S. Tang, B.-S. Lee, and B. He, "Fair resource allocation for data-intensive computing in the cloud," *IEEE Trans. Services Computing*, 2016, in press.
[45] E. Hwang and K. H. Kim, "Minimizing cost of virtual machines for deadline-constrained mapreduce applications in the cloud," in *Proceedings of the ACM/IEEE 13th International Conference on Grid Computing*. IEEE Computer Society, 2012, pp. 130–138.
[46] L. Gu, D. Zeng, S. Guo, Y. Xiang, and J. Hu, "A general communication cost optimization framework for big data stream processing in geo-distributed data centers," *IEEE Trans. Computers*, vol. 65, no. 1, pp. 19–29, 2016.
[47] K. Chen, J. Powers, S. Guo, and F. Tian, "Cresp: Towards optimal resource provisioning for mapreduce computing in public clouds," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1403–1412, 2014.
[48] D. Cheng, P. Lama, C. Jiang, and X. Zhou, "Towards energy efficiency in heterogeneous hadoop clusters by adaptive task assignment," in *IEEE ICDCS'15*. IEEE, 2015, pp. 359–368.
[49] K. K. Nguyen and M. Cheriet, "Environment-aware virtual slice provisioning in green cloud environment," *IEEE Trans. services computing*, vol. 8, no. 3, pp. 507–519, 2015.
[50] S. M. Nabavinejad and M. Goudarzi, "Energy efficiency in cloud-based mapreduce applications through better performance estimation," in *DATE'16*. IEEE, 2016, pp. 1339–1344.
[51] Y. Shao, C. Li, J. Gu, J. Zhang, and Y. Luo, "Efficient jobs scheduling approach for big data applications," *Computers & Industrial Engineering*, vol. 117, pp. 249–261, 2018.
[52] T. Gouasmi, W. Louati, and A. H. Kacem, "Cost-efficient distributed mapreduce job scheduling across cloud federation," in *IEEE SCC'17*. IEEE, 2017, pp. 289–296.
[53] J. W. Park, A. Tumanov, A. Jiang, M. A. Kozuch, and G. R. Ganger, "3sigma: distribution-based cluster scheduling for runtime uncertainty," in *EuroSys*. ACM, 2018, p. 2.
[54] M. Mazzucco and M. Dumas, "Reserved or on-demand instances? a revenue maximization model for cloud providers," in *IEEE CLOUD'11*. IEEE, 2011, pp. 428–435.

Seyed Morteza Nabavinejad is a Post-Doctoral Research Fellow at the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. He received the B.Sc. degree in Computer Engineering from Ferdowsi University of Mashhad and the M.Sc., and Ph.D. degrees from Sharif University of Technology in 2011, 2013, and 2018, respectively. He has published several peer reviewed papers in venues such as *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Big Data*, *DATE*, and *SAC*. His research interests include big data processing, cloud computing, green computing, approximate computing, and energy aware data centers

Maziar Goudarzi (S00, M11, SM'16) is an Associate Professor at the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. He received the B.Sc., M.Sc., and Ph.D. degrees in Computer Engineering from Sharif University of Technology in 1996, 1998, and 2005, respectively. Before joining Sharif University of Technology as a faculty member in September 2009, he was a Research Associate Professor at Kyushu University, Japan from 2006 to 2008, and then a member of research staff at University College Cork, Ireland in 2009. His current research interests include architectures for large-scale computing systems, green computing, hardware-software codesign, and reconfigurable computing. Dr. Goudarzi has won two best paper awards, published several papers in reputable conferences and journals, and served as member of technical program committees of a number of IEEE, ACM, and IFIP conferences including ICCD, ASP-DAC, ISQED, ASQED, EUC, and IEDEC among others.